Vortrag im Seminar aus maschinellem Lernen

Über das Paper:

Mining Concept-Drifting Data Streams using Ensemble Classifiers

Haixun Wang, Wei Fan, Philip S. Yu, Jiawei Han

Vortrag: Robert Deußer

Gliederung

- Einführung
- Ensemble Classifier
  - Idee
  - Fehlerreduzierung durch Ensemble
- Einsatz
  - Gewichtsanpassung
  - Algorithmus
  - (Instance Based) Ensemble Pruning
- Ergebnisse
- Diskussion
- (Exkurs: Inkrementelle Regellerner)

Einleitung

- Datenströme, Item-Sets
- Inkrementelle Lernverfahren (VFDT, …)
- Blockweise Batch (k-means)
- Topic Drift
- Problem: Up-to-date vs. Outdated concepts
- Herausforderungen beim Lernen in Datenströme (mit Topic Drift):
  - Accuracy
  - Efficiency
  - Ease of Use

(Weighted) Classifier Ensembles

- Idee: Trainiere *unterschiedliche* Classifier und klassifiziere Beispiele via *Abstimmung*
- Hier:
  - Teile Datenstrom in Blöcke fester Länge (chunks)
  - Trainiere auf jedem Block einen Classifier
  - Bei Abstimmung hat jeder Classifier ein Gewicht, das umgekehrt proportional zu seinem vermuteten Fehler ist

Fehlerreduzierung durch Ensemble

- Sei C_i der Classifier, der auf Block S_i trainiert wurde, E_k das Ensemble bestehend aus den letzten k trainierten Classifiern und G_k ein Classifier der auf den letzten k Blöcken trainiert wurde.
- Dann hat das (umgekehrt proportional zum Klassifikationsfehler gewichtete) Classifier Ensemble E_k Fehler <= dem Fehler des Classifiers G_k

- Anmerkung: G_j mit $j < k$ kann trotzdem genauer sein als E_k

Gewichtsanpassung

- Gesucht: Gewichte die umgekehrt proportional zum Klassifikationsfehler sind
- Kann Klassifikationsfehler nur abschätzen
- Annahme: Aktueller Block S_n kommt dem zu klassifizierenden Beispiel am nächsten (Problem?)
- MSE als Maß für den Klassifikationsfehler:
  - MSE_r: MSE für Classifier, der a-priori W-keit zur Klassifikation nimmt

$$MSE_r = \sum_c p(c)(1 - p(c))^2$$

  - MSE_i: MSE für Classifier C_i

$$MSE_i = \frac{1}{|S_n|} \sum_{(x,c) \in S_n} (1 - f_c^i(x))^2$$

  - Das Gewicht von C_i ist dann

$$w_i = MSE_r - MSE_i$$

Gewichtsanpassung (2)

- Analog wenn Kosten berücksichtigt werden sollen
- Benefitmatrix:

	predict $fraud$	predict $\neg fraud$
actual $fraud$	$t(x) - cost$	0
actual $\neg fraud$	$-cost$	0

- Benefit wenn *x*, das zu Klasse *c* gehört als zu Klasse *c'* gehörend klassifiziert wird: $b_{c,c'}(x)$
- Dann ist Benefit für Classifier C_i: $b_i = \sum_{(x,c) \in S_n} \sum_{c'} b_{c,c'}(x) \cdot f_{c'}^{i}(x)$
- Das Gewicht von C_i ist dann: $w_i = b_i - b_r$

Algorithmus

Input: S: a dataset of `ChunkSize` from the incoming stream
K: the total number of classifiers
$\mathcal{C}$: a set of K previously trained classifiers

Output: $\mathcal{C}$: a set of K classifiers with updated weights

train classifier $\mathcal{C}'$ from S;
compute error rate / benefits of $\mathcal{C}'$ via cross validation on S;
derive weight w' for $\mathcal{C}'$ using (8) or (9);
for *each classifier* $\mathcal{C}_i \in \mathcal{C}$ **do**
  apply $\mathcal{C}_i$ on S to derive MSE_i or b_i;
  compute w_i based on (8) and (9);
$\mathcal{C} \leftarrow K$ of the top weighted classifiers in $C \cup \{C'\}$;
return $\mathcal{C}$;

- Komplexität: $O(p \cdot f(n/p) + Kn)$

  n: Anzahl Elemente im Datenstrom

  p: Anzahl Partitionen (n/p = Blockgröße)

  f(s): Aufwand einen Classifier auf s zu trainieren (i.d.R. super-linear)

(Ensemble) Pruning

- Problem: Alle *K* Classifier werden zum Klassifizieren konsultiert
- Ziel: Eine Teilmenge von *C* (den gelernten Classifiern), die *ähnlich gut* wie *C* ist, finden
- Ansätze:
  - Finden einer Teilmenge C‘ von C, die kleinstmöglichen MSE hat
  - Finden einer Teilmenge C‘ von C die größtmöglichen Abstand (KL-distance) hat
- Problem: Aufwand, Cost-sensitive Applications(?), Topic Drift (bei KL-distance)

Instance based Pruning (IBP)

- Es werden in der Regel nicht alle *K* Classifier benötigt
- Es genügt eine ausreichend sichere Vorhersage zu haben
- Bei ausreichend sicherer Vorhersage, werden keine weiteren Classifier befragt
- Setzt eine gewisse Fehlertoleranz der Anwendung voraus

$$p(\text{fraud} \mid y) \cdot t(y) > \text{cost} \Leftrightarrow \; p(\text{fraud} \mid y) > \text{cost} / t(y)$$

Beispiel:

t(y) = 900$, cost = 90$

Sobald sicher ist das *p(*fraud | *y)* > 0.1 wird Untersuchung veranlasst

Instance based pruning (2)

- Die gewichtete W-keit nach konsultieren von *k* Classifiern ist

$$F_k(x) = \frac{\sum_{i=1}^{k} w_i \cdot p_i(\text{fraud} \mid x)}{\sum_{i=1}^{k} w_i}$$

- Mit einem Fehler $\varepsilon_k(x) = F_k(x) - F_K(x)$

- Einteilen des Wertebereichs von $F_k(\cdot)$ ([0,1]) in *z* Körbe (bins), ein Beispiel *x* gehört zu bin *(i, k)* wenn

$$F_k(x) \in [\frac{i}{z}, \frac{i+1}{z})$$

- Für jeden bin berechnen von

$\mu_{k,i}$: durchschnittlicher Fehler aller Beispiele

$\sigma^2_{k,i}$: Varianz des Fehlers aller Beispiele

IBP - Klassifikation

- Überprüfen ob Klassifikation sicher ist (*i* ist bin zu dem *y* gehört)

$$\begin{cases} F_k(y) - \mu_{k,i} - \mathtt{t} \cdot \sigma_{k,i} > cost/t(y), & \text{fraud} \\ F_k(y) + \mu_{k,i} + \mathtt{t} \cdot \sigma_{k,i} \leq cost/t(y), & \text{non-fraud} \\ \text{otherwise}, & \text{uncertain} \end{cases} \quad (10)$$

- Algorithmus:

Input: y: a test example
$\mathtt{t}$: confidence level
$\mathcal{C}$: a set of K previously trained classifiers
Output: prediction of y's class

Let $\mathcal{C} = \{\mathcal{C}_1, \cdots, \mathcal{C}_n\}$ with $w_i \geq w_j$ for $i < j$;
$F_0(y) \leftarrow 0$;
$w \leftarrow 0$;
for $k = \{1, \cdots, K\}$ **do**
  $F_k(y) \leftarrow \frac{F_{k-1} \cdot w + w_k \cdot p_k(fraud|x)}{w + w_k}$;
  $w \leftarrow w + w_k$;
  let i be the bin y belongs to;
  apply rules in (10) to check if y is in $\mathtt{t}$-σ region;
  return fraud/non-fraud if $\mathtt{t}$-σ confidence is reached;
if $F_K(y) > cost/t(y)$ **then**
  return fraud;
else
  return non-fraud;

IBP - Wertebestimmung

- Durchschnittlicher Fehler und Varianz können beim Trainieren des Classifiers ermittelt werden

Input: S: a dataset of `ChunkSize` from the incoming stream
K: the total number of classifiers
ξ: number of bins
$\mathcal{C}$: a set of K previously trained classifiers
Output: $\mathcal{C}$: a set of K classifiers with updated weights
μ, σ: mean and variance for each stage and each bin

...

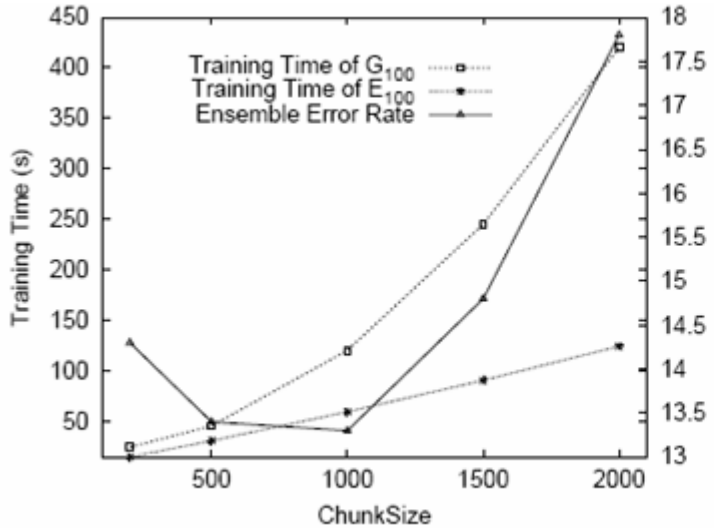
$\mathcal{C} \leftarrow K$ of the top weighted classifiers in $\mathcal{C} \cup \{\mathcal{C}'\}$;
for *each* $y \in S$ **do**
compute $F_k(y)$ for $k = 1, \cdots, K$;
y belongs to bin (i, k) if $F_k(y) \in [\frac{i}{\xi}, \frac{i+1}{\xi})$;
incrementally updates $\mu_{i,k}$ and $\sigma_{i,k}$ for bin (i, k);

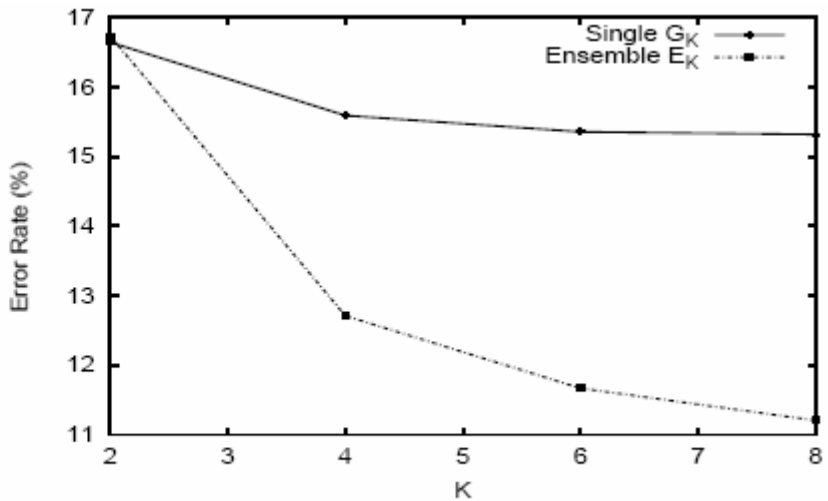
- Aufwand bleibt gleich: $O(p \cdot f(n/p) + Kn)$

Ergebnisse – Synthetischer DS

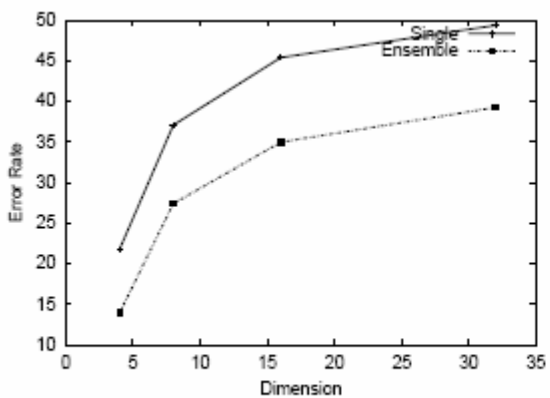
- Beispiele gleichverteilt in $[0,1]^d$
- Eine Hyperebene teilt den Beispielraum in zwei Teile gleich großen Volumens (pos/neg)
- Um Topic Drift zu simulieren, werden die Dimensionsgewichte der Hyperebene kontinuierlich verändert
- Parameter:
  - k: Wie viel Dimensionen ändern ihr Gewicht = 20%
  - N: Wie oft findet einer Änderung statt, alle N = 1000 Beispiele, 10% Chance das sich das Vorzeichen der Änderung umdreht
  - t: Größe der Änderung = 0.1
  - Noise: Bei 5% aller Beispiele werden die Labels getauscht

Ergebnisse – Synthetischer DS

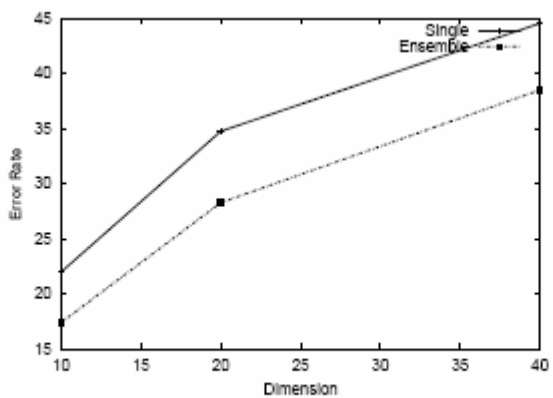




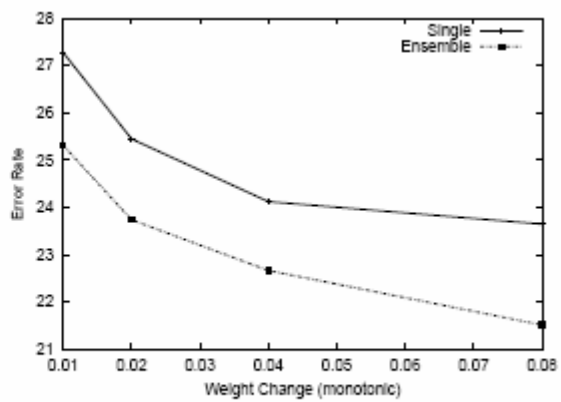
(a) Varying window size/ensemble size



(a) # of changing dimensions



(b) total dimensionality

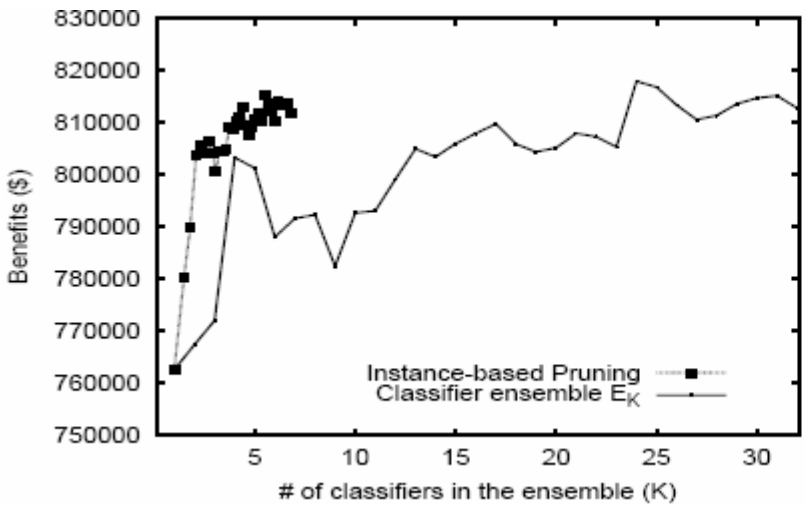


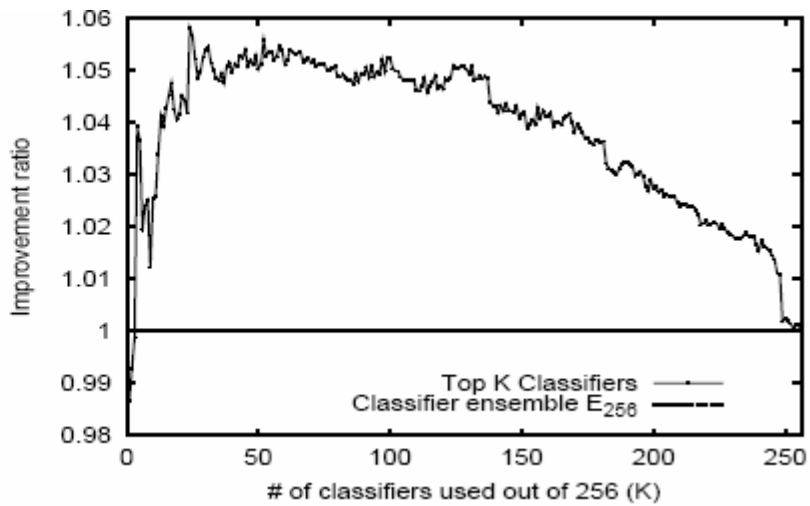
(c) (monotonic) weight change per dimension

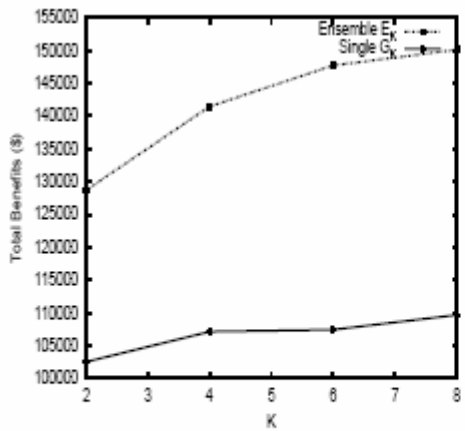
Ergebnisse – Credit Card Data

- Aus „Real life“ Daten erhalten
- 5 Millionen Datensätze, aus dem Zeitraum eines Jahres
- Zwei Datenströme
  - Chronologisch sortiert
  - Nach Transaktionswert sortiert
- Parameter:
  - cost (für die Untersuchung einer Transaktion): 90$
  - benefit: Gerettete Summe - Kosten aller Untersuchungen

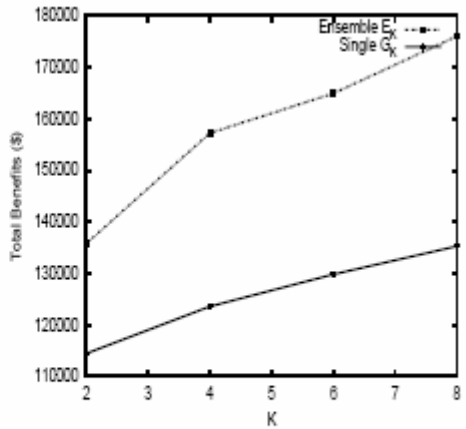
Ergebnisse – Credit Card Data







(a) Varying K (original stream)



(c) Varying K (simulated stream)

Zusammenfassung

- Ensemble Classifier haben gegenüber herkömmlichen Classifiern die folgenden Vorteile
  1. Verbesserung der Klassifikationsgenauigkeit
  2. Modell wird Effizienter gelernt
  3. Lassen sich gut Skalieren / Parallelisieren
  4. Berücksichtigung von Topic Drift
- Instance based Pruning kann bei geeigneter Problemstellung den Klassifikationsaufwand erheblich reduzieren

Diskussion & Fragen

(Exkurs) Inkrementelle Regellerner

- „Discovering Decision Rules from Numerical Data Streams“
- „Incremental Rule Learning based on Example Nearness from Numerical Data Streams“
  F. Ferrer-Troyano, J. Aguilar-Ruiz, J. Riquelme

- Adressiert werden ähnliche Ziele (effizientes Lernen in Datenströme mit *Topic Drift*), aber Fokus auf HighSpeed Datenströme
- Entwickelt für Lernen in Datenströmen
- Autorenteam hat zwischen 2001 und 2006 vier Papers zu dem Themenkomplex veröffentlicht
- Algorithmen werden im Detail beschrieben, Begründungen für das Funktionieren, den Aufwand etc. werden nicht gegeben. Paper sind aufgrund schlechtem Englisch schwer verständlich

Zusammenfassung

- Regeln bestehen aus *d* Intervallen, eine Regel ist dann die Konjunktion aller *d* Intervallwerte
- Regeln für verschiedene Klassenlabel überlappen nicht
- Klassifikation: Wenn Regel das Beispiel überdeckt ist es klassifiziert, ansonsten Votingverfahren
- Training: Jede Klasse wird durch *a* Regeln beschrieben, wenn neues Beispiel nicht abgedeckt wird, wächst eine Regel um es einzuschliessen (das geringste Wachstum wird genommen)
- Beim ersten Algorithmus wird das Wachstum von Regeln durch Growth-Bounds eingeschränkt
- Beim zweiten Algorithmus werden bei Entscheidungsgrenzen unsaubere Regeln (überdecken pos/neg Beispiele) zugelassen und bei Bedarf getrennt